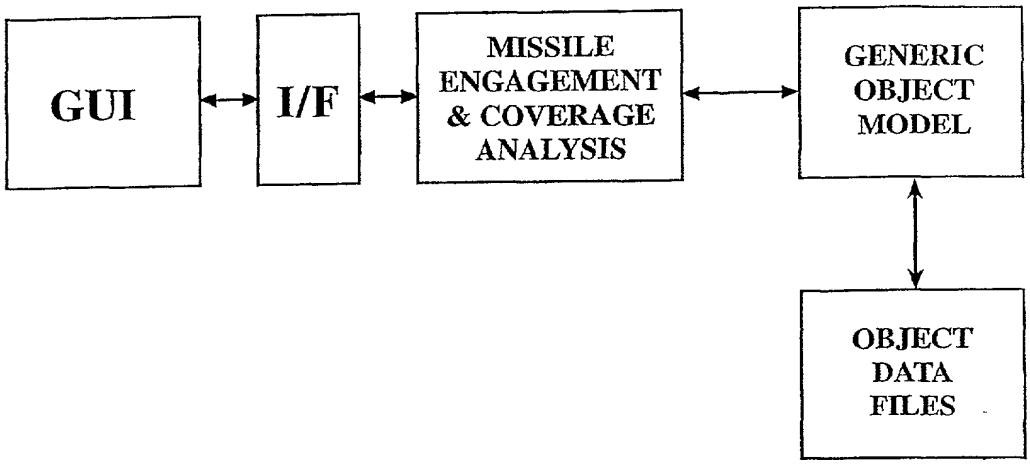
GUI
I/F
MISSILE ENGAGEMENT & COVERAGE ANALYSIS
GENERIC OBJECT MODEL
OBJECT DATA FILES

Fig 1

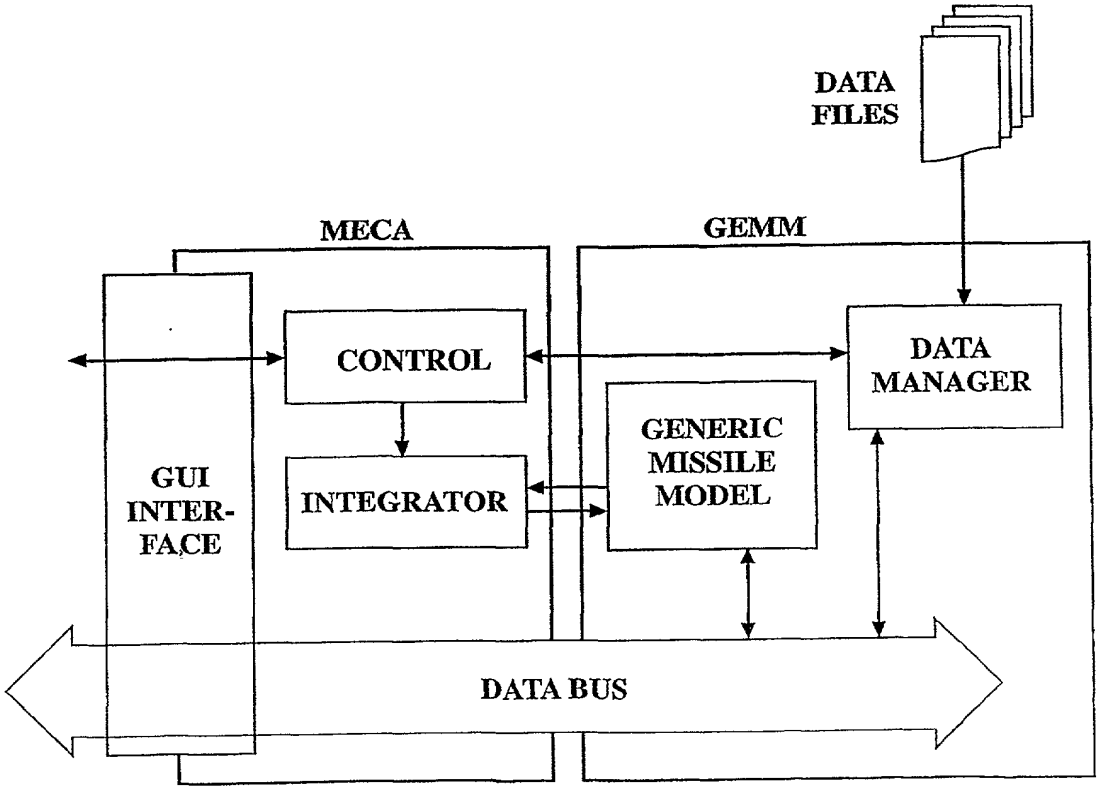
DATA FILES
MECA
GEMM
CONTROL
DATA MANAGER
GENERIC MISSILE MODEL
GUI INTER-FACE
INTEGRATOR
DATA BUS

FIG 2

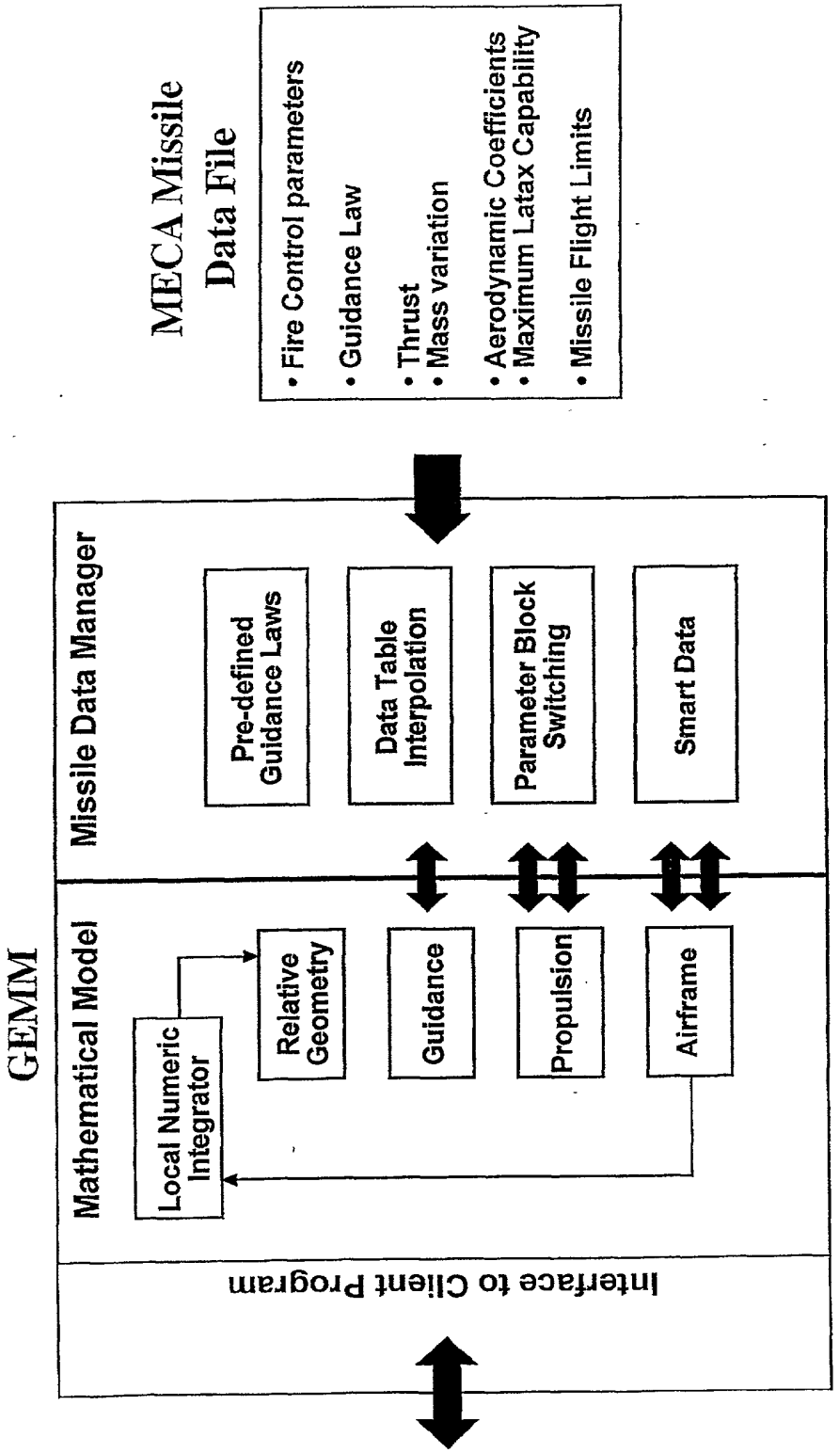

Fig 3

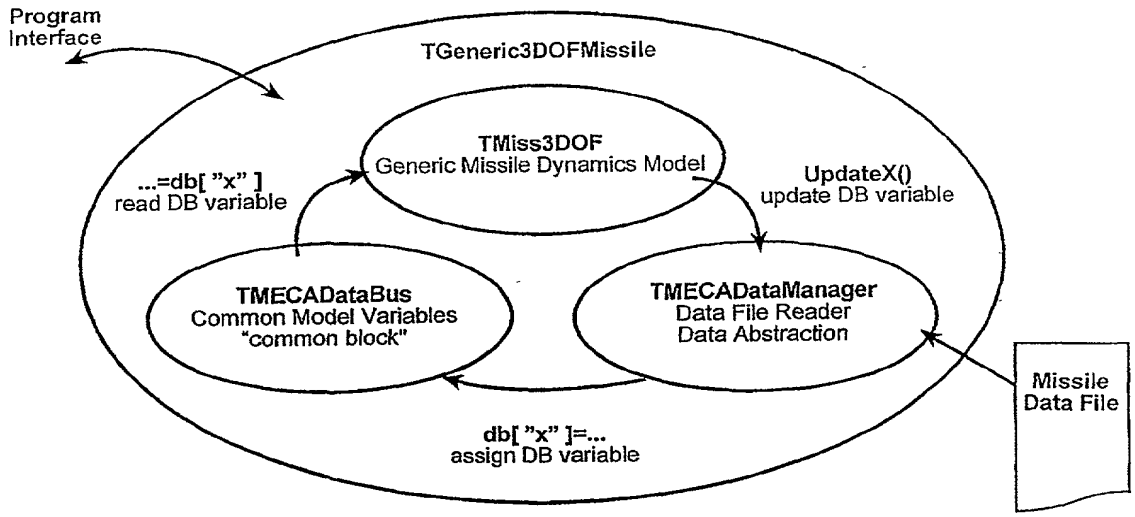
Program
Interface
TGeneric3DOFMissile
TMiss3DOF
Generic Missile Dynamics Model
...=db["x"]
read DB variable
UpdateX()
update DB variable
TMECADataBus
Common Model Variables
"common block"
TMECADataManager
Data File Reader
Data Abstraction
Missile
Data File
db["x"]=...
assign DB variable

Fig 4

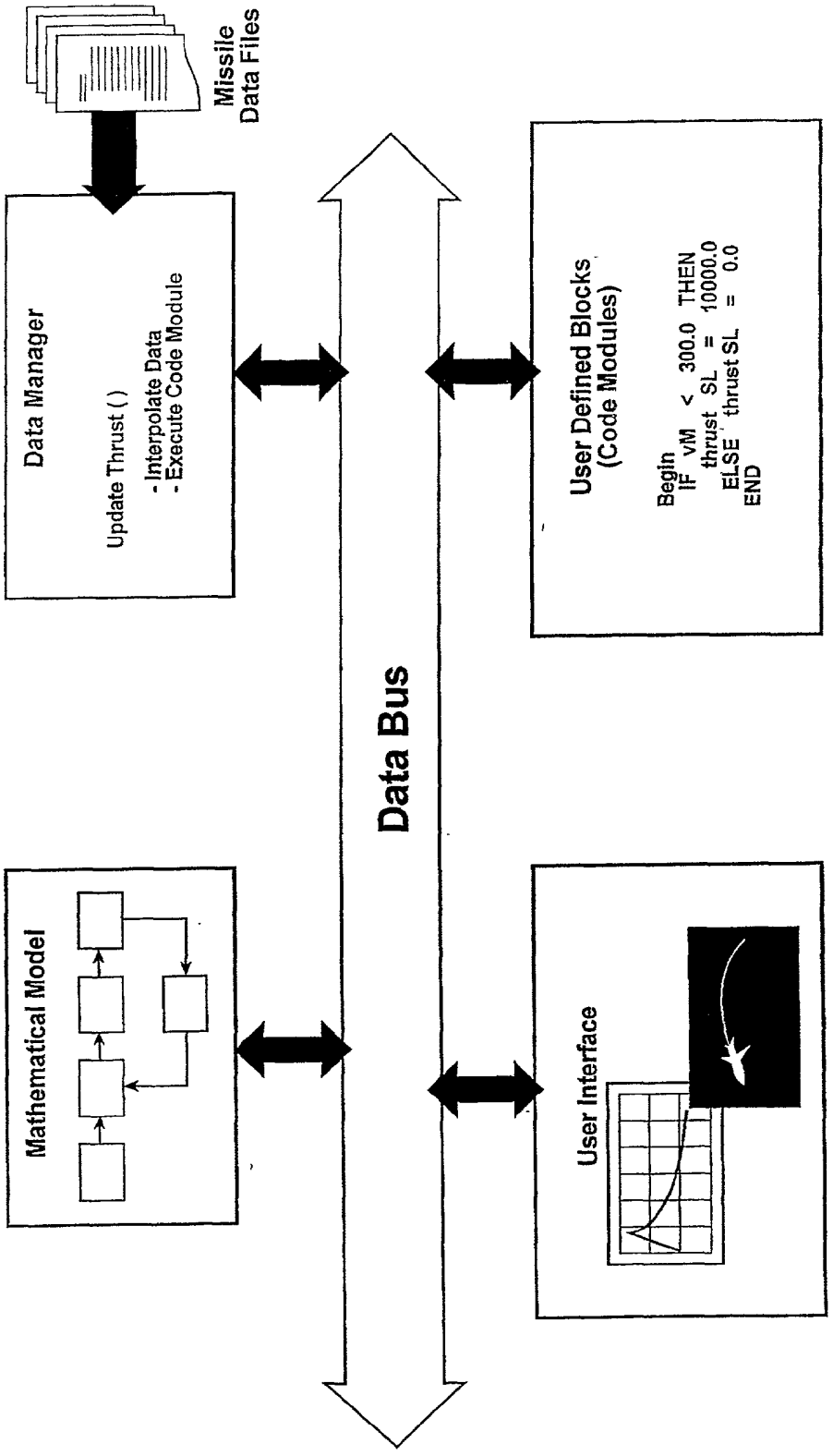
Missile
Data Files
Data Manager
Update Thrust ( )
- Interpolate Data
- Execute Code Module
User Defined Blocks
(Code Modules)
Begin
IF vM < 300.0 THEN
thrust SL = 10000.0
ELSE thrust SL = 0.0
END
Data Bus
Mathematical Model
User Interface

Fig 5

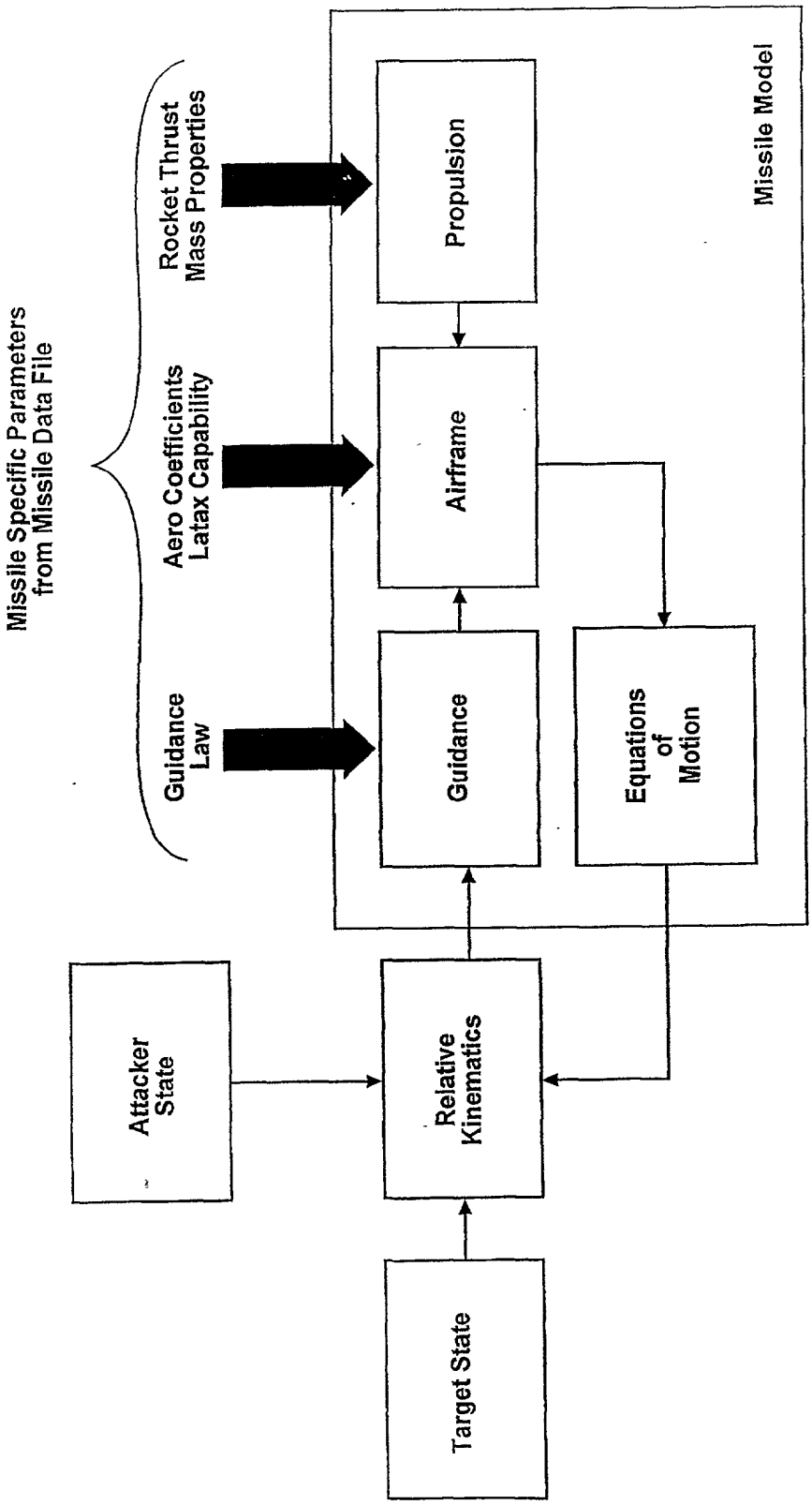
Missile Specific Parameters from Missile Data File
Rocket Thrust Mass Properties
Aero Coefficients Latax Capability
Guidance Law
Propulsion
Airframe
Guidance
Equations of Motion
Missile Model
Attacker State
Relative Kinematics
Target State

Fig 6

Execution Process:
1. Data manager loads missile file, parsing it into an internal representation.
2. Data manager updates initial state of data bus variables and internal model options.
3. Model executes for each simulation time-step, calling user defined module where necessary via the StackMac virtual machine.

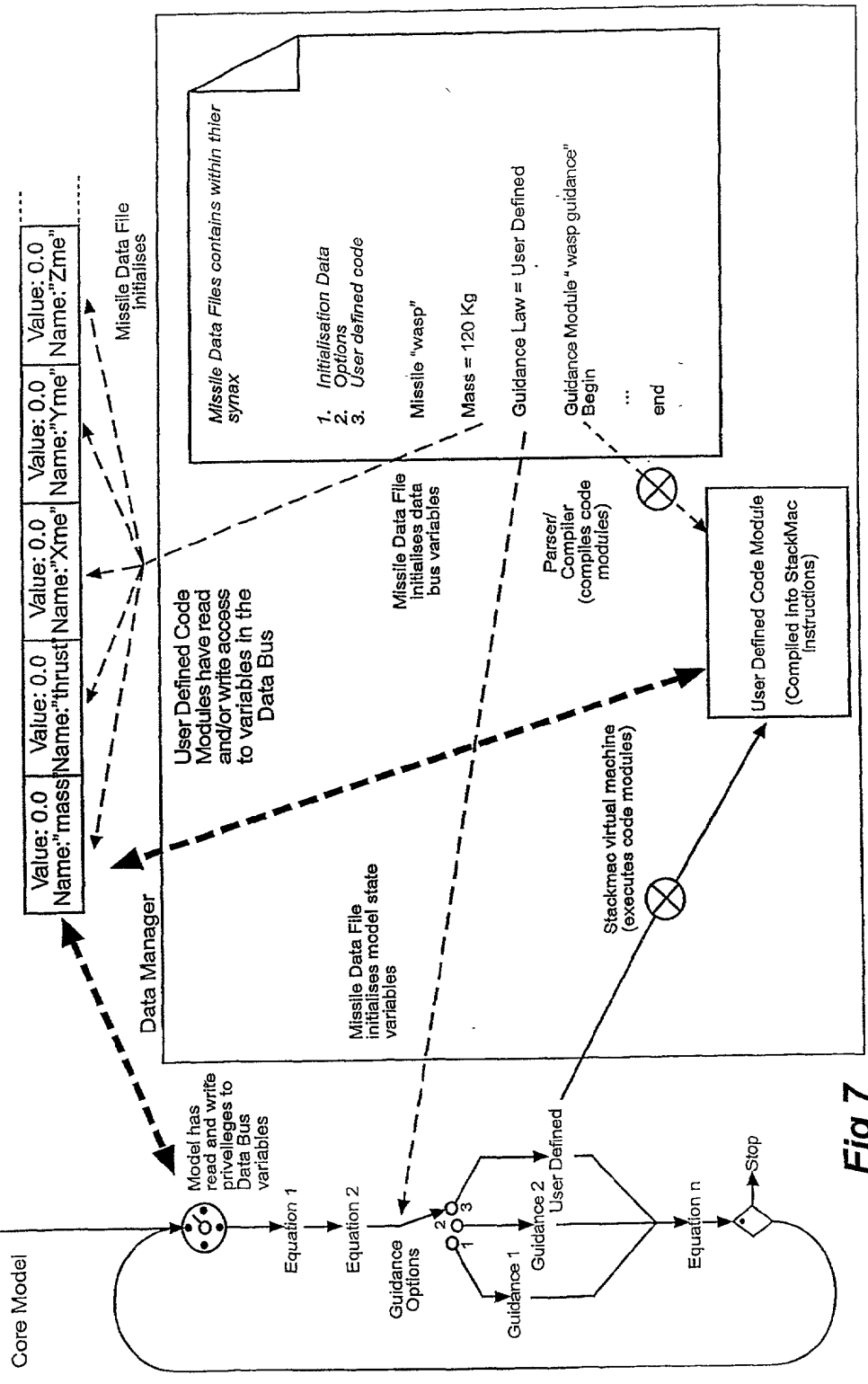

Fig 7

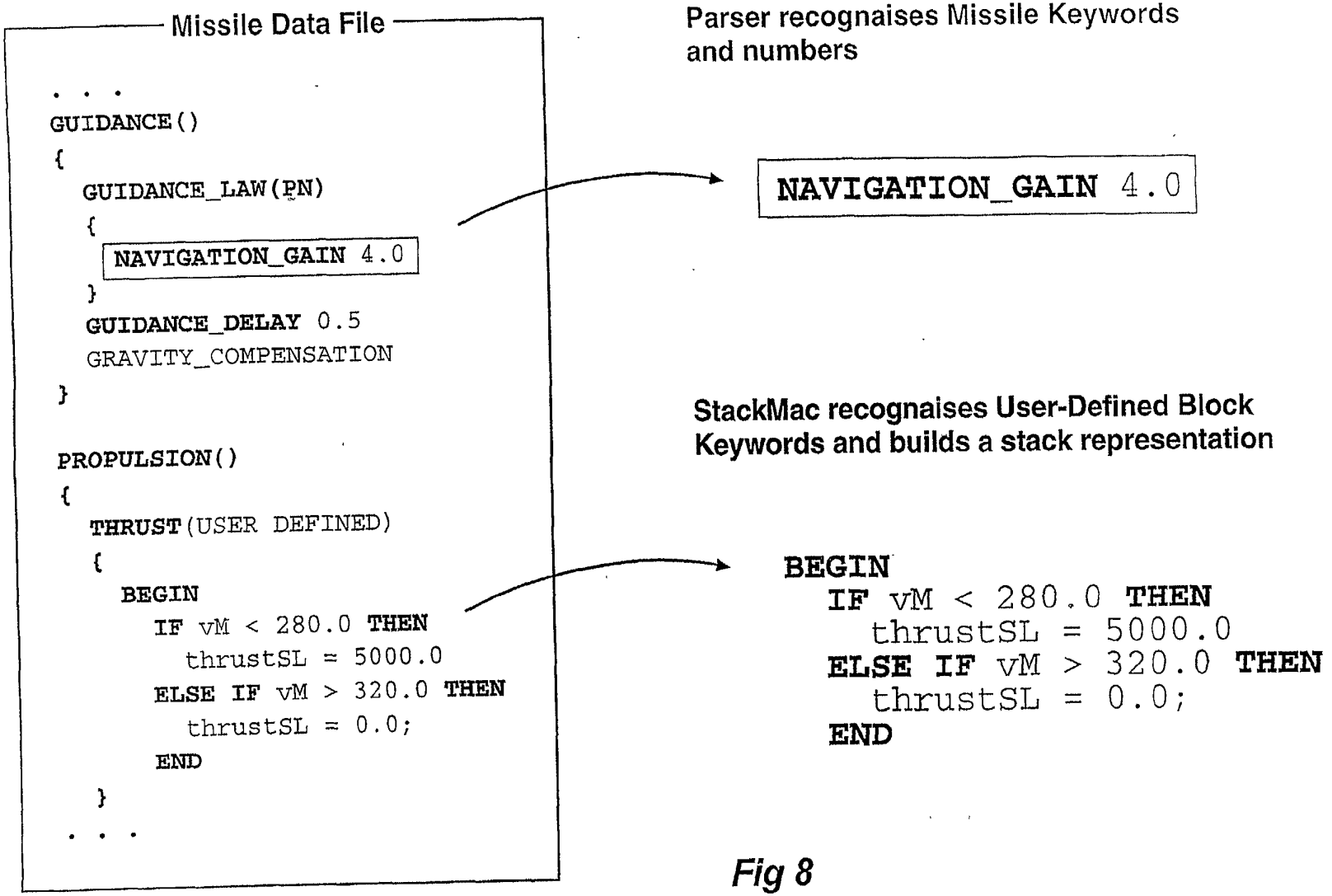
Missile Data File
. . .
GUIDANCE()
{
GUIDANCE_LAW(PN)
{
NAVIGATION_GAIN 4.0
}
GUIDANCE_DELAY 0.5
GRAVITY_COMPENSATION
}
PROPULSION()
{
THRUST(USER DEFINED)
{
BEGIN
IF vM < 280.0 THEN
thrustSL = 5000.0
ELSE IF vM > 320.0 THEN
thrustSL = 0.0;
END
}
. . .
Parser recognaises Missile Keywords and numbers
NAVIGATION_GAIN 4.0
StackMac recognaises User-Defined Block Keywords and builds a stack representation
BEGIN
IF vM < 280.0 THEN
thrustSL = 5000.0
ELSE IF vM > 320.0 THEN
thrustSL = 0.0;
END

Fig 8

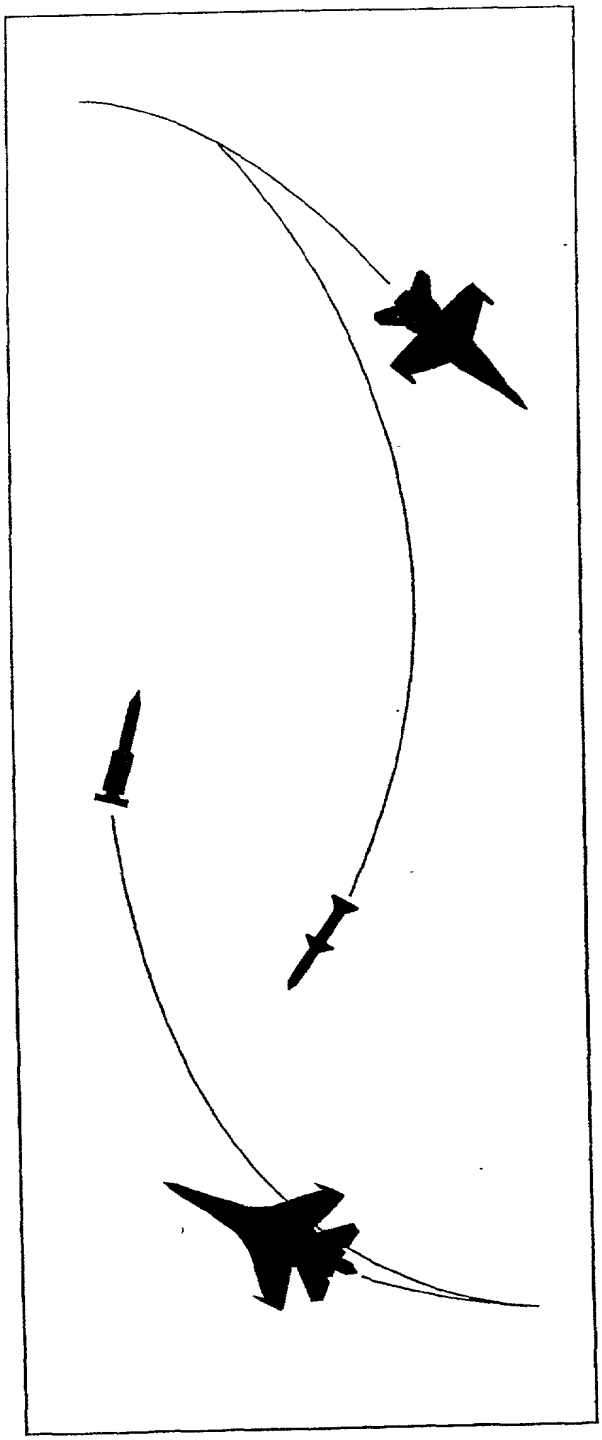

Fig 9

Programstartup

```
// Create Engagement Objects
TEngagement model;
...
```

```
// Set Attacker Parameters
model.SetAttackerPosition(x,y,z)
model.SetAttackerSpeed(v);
model.SetAttackerHeading(psi,theta);
...
// Set model time step
model.SetTimeStep(0.1);
...

// Run Engagement model
while (result != FINISHED)
{
 result = model.Step();
}
```

TABLE 1

Programstartup

```
// Create Aircraft Objects
TAircraft redAircraft;
TAircraft blueAircraft;


// Create Missile  Objects
TMissile redMissile;
TMissile blueMissile;
...

// Load Missile Data Files
redMissile.Load("red.mis")
blueMissile.Load("blue.mis")
...
```

```
// Run Simulation
while (result != FINISHED)
{
   // Propagate aircraft
   redAircraft.Propagate();
   blueAircraft.Propagate();

   // Propagate missiles
   if (redLaunched)
   redMissile.Propagate(blueAircraft);

   if (blueLaunched)
   blueMissile.Propagate(redAircraft);

   // Check simulation STOP logic
   result = Checkstop();
   }
   ...
```

TABLE 2